JOURNAL OF COMPUTER AND SYSTEM SCIENCES 37, 144–155 (1988)

$k+1$ Heads Are Better than k for PDAs

MAREK CHROBAK*

Institute of Informatics, Warsaw University, PKiN VIIIp., 00-901 Warsaw, Poland

AND

MING LI†

Aiken Computation Laboratory, Harvard University, Cambridge, Massachusetts 02138

Received October 7, 1987

We prove the following conjecture stated by Harrison and Ibarra in (*Inform. and Control* **13** (1968), 462): There are languages accepted by $(k+1)$-head 1-way deterministic pushdown automata ($(k+1)$-DPDA) but not by k-head 1-way pushdown automata (k-PDA), for every k. On the assumption that their conjecture holds, Harrison and Ibarra also derived some other consequences. Now all those consequences become theorems. For example, the class of languages accepted by k-PDAs is not closed under intersection and complementation. Several other interesting consequences also follow: CFL $\not\subseteq \bigcup_k$ DPDA(k) and FA(2) $\not\subseteq \bigcup_k$ DPDA(k), where DPDA(k) = $\{L \mid L$ is accepted by a k-DPDA$\}$ and FA(2) = $\{L \mid L$ is accepted by a 2-head FA$\}$. Our proof is constructive (that is, not based on diagonalization). Before, the "$k+1$ versus k heads" problem was solved by diagonalization and translation methods for stronger machines (2-way, etc) and by traditional counting arguments for weaker machines (k-FA, k-head counter machines, etc).

1. INTRODUCTION

From the 1960s when formal language and automata theory flourished, two open questions in automata and formal language theory were raised:

(1) Rosenberg conjecture (1965): $(k+1)$-head 1-way finite automata are better than k-head 1-way finite automata [R1, R2].

* Current address: Department of Mathematics and Computer Science, University of California, Riverside, CA 92521.

† Work supported by NSF Grant DCR-8606366 at Ohio State University, ONR Grant N00014-85-k-0445 at Harvard University, and Ohio State University Seed Grant. On leave from Department of Computer and Information Science, Ohio State University, Columbus, OH 43210.

0022-0000/88 $3.00

(2) Harrison–Ibarra conjecture (1968): $(k+1)$-head 1-way pushdown automata are better than k-head 1-way pushdown automata. Or, there are languages accepted by $k+1$-DPDA but no k-PDA [HI].

One of the major problems in the theory of automata and complexity is to determine whether additional computational resources (heads, stacks, tapes, etc.) increase the computational power of the investigated machine. In the case of multihead machines it is natural to ask whether $k+1$ heads are better than k. In 1965, Rosenberg [R2] claimed a solution to problem (1), but Floyd [F] pointed out that Rosenberg's informal proof was incomplete. In FOCS '71 Sudborough [S1, S2], and later Ibarra and Kim [IK], obtained a partial solution to problem (1) for the case of 2 heads versus 3 heads. (Sudborough's result was also for writing automata.) In the 1976 FOCS meeting, Yao and Rivest [YR] finally presented a full solution to problem (1). A different proof was also obtained by Nelson [N].

Problem (2), however, is still open. Several authors tried to generalize the Yao–Rivest method [M1, M2] or the Ibarra–Kim method [C] to the k-PDA case, but only partial results were obtained. We list some related results and partial solutions regarding question (2):

(a) In 1968, Harrison and Ibarra [HI, I1] presented a class of languages and suggested that these languages can separate the k-PDA hierarchy which would answer their conjecture in (2). (Their languages need a simple extension as we shall see.) They also proved many important consequences provided their conjecture holds.

(b) In 1973, Ibarra [I2] showed that (2) is true for 2-way DPDAs and PDAs and $k+2$ heads are better than k for DFAs by diagonalization and translational methods, using the simulation results in [AHU, C1]. For 2-way finite automata the problem was solved by Seiferas [S, S0] and Monien [M3, M4].

(c) In 1982, Miyano [M1] showed that question (2) is true if the pushdown store is replaced by a counter. He generalized the argument of [YR].

(d) In 1983, Miyano [M2] showed that if the input is not bounded by end markers then question (2) is true. Again, he relaxed the condition so that the diagonalization is possible. Combining results in (c) and (d), Miyano observed in [M2] that question (2) is hard because "the 1-way k-head PDA's are too complex to be analyzed by the counting arguments (as used in [YR]) and, on the other hand, they are not so powerful to allow the diagonalization" (as used in [12]).

(e) A major progress was made by Chrobak [C] in 1985 who showed that question (2) is true for the deterministic case. However, it was observed in [C] that the method used there does not work for the general (nondeterministic) case. Further, the method in [C] was not constructive. That is, no specific separation language could be constructed (only the existence is proved). The last drawback of the method in [C] is that the proof is terribly long and complicated.

In this paper we will give a complete and transparent solution to the

Harrison–Ibarra conjecture for the general case. The proof is completely different from [M1, M2, C]. It not only settles the conjecture, it is also constructive and much simpler than the partial solution in [C]. Several consequences derived in [HI] based on the conjecture now become true. We also show that there is a language which is accepted by a 2-FA but not k-DPDA for any k. Further, there is a context-free language that is not acceptable by k-DPDA for any k.

One of the major concerns in computational complexity theory is to develop lower bound techniques without using diagonalization (because of the oracle results [HU]). In this paper we attack such a problem, as mentioned in (d) above, which sits in between the conventional counting arguments and the diagonalization proofs.

Informally, a k-head 1-way pushdown automaton, denoted as k-PDA, is a PDA (pushdown automaton [HU]) with k 1-*way* read heads on the input tape and a pushdown store (stack for short). We write k-DPDA to denote the deterministic version of k-PDA. We also write k-FA (k-DFA) to denote k-head 1-way (deterministic) finite automata. We simply assume that the heads can "see" each other (the theorems still hold if they do not). Each step, some heads may move to the right and the machine may change its state, depending on the k symbols read by the k heads, the current state, and the top stack symbol. The machine accepts (and halts) by emptying the stack and entering a final state (see [HI]). The input is enclosed by two end markers. When all heads reach the second end marker the machine halts. By a configuration of a k-PDA at time t, denoted shortly by ID_t, we mean a tuple: (positions of input heads, state of the machine, top stack symbol, height of stack).

We refer the readers to [HI] for formal definitions and the significance of k-PDAs. We will use PDA(k) (DPDA(k), FA(k), DFA(k)) to represent the class of languages accepted by k-PDA (k-DPDA, k-FA, k-DFA).

2. Main Result: The Hierarchy Theorem

We prove our major theorem in this section. The following language was basically defined by Rosenberg [R] in order to show that $k+1$ heads are better than k heads for finite automata:

$$L_b = \{w_1 \# \cdots \# w_b \$ w_b \# \cdots \# w_1 \mid w_i \in \{0, 1\}^*\}.$$

Although Rosenberg's proof was incomplete, the proof by Yao and Rivest depended on this language [YR]. Amazingly, this language can also be used to serve our purpose although the proof technique is new. Actually the language proposed by [HI] is the same except that [HI] required $b = k$, whereas [R1, R2, YR] required $b = \binom{k}{2}$.

Our proof also used Kolmogorov-complexity (K-complexity) which is defined as below. Fix some standard enumeration of Turing machines. The K-complexity of a

string x, denoted $K(x)$, is the length of the shortest program that prints x. A string x is *random* if $K(x) \geq |x|$. The conditional K-complexity of x with respect to y, denoted by $K(x|y)$, is the length of the shortest program which, with input y, prints x. We state two simple well-known facts without proof (see [PSS]).

FACT 1. *There exist random strings of every length.*

FACT 2. *If a string uvw is random, then* $K(v|uw) \geq |v| - O(\log |uvw|)$.

The use of K-complexity for the lower bound proofs was first introduced in [P] and then in [PSS] and later by many other authors [P1, RS, PS, M5, MS, LV, GKS, LLV]. Our proof presents another novel application of K-complexity.

MAIN THEOREM. *L_b can be accepted by a k-PDA if and only if* $b \leq \binom{k}{2}$.

Proof. Since, by [R2] and [YR], L_b can be accepted by even a k-DFA when $b \leq \binom{k}{2}$, we need only to prove that if $b > \binom{k}{2}$ then L_b cannot be accepted by a k-PDA.

Assume $b > \binom{k}{2}$. First we need the following technical fact.

FACT 3. *If M is a k-PDA then there is an equivalent k-PDA M' such that each computation of M' has the following property: whenever M' increases the stack, then it does not decrease it until one of the heads moves forward. In other words, M' does not make up–down reversals on the stack in time intervals when its heads are stationary.*

Proof. An elementary exercise in automata theory. ∎

From now on we simply assume that M satisfies the condition of the above fact. Now suppose that a k-PDA M accepts L_b. Choose a very long random string $W \in \{0, 1\}^n$, where $n = b\alpha(k^2+1)$ and

$$K(W) \geq |W|,$$

for α large enough, so that all the subsequent formulas make sense. Equally divide W into b *blocks* $W = w_1 w_2 \cdots w_b$. Further equally divide each w_i into $m = k^2+1$ *sub-blocks* $w_i = w_{i1} w_{i2} \cdots w_{im}$. All w_{ij}'s have length α. Construct input

$$I = \text{¢} w_1 \# \cdots \# w_b \$ w_b \# \cdots \# w_1 \text{¢},$$

where ¢'s are end markers. M should accept I. Fix a *shortest* accepting computation COMP of M on I. Let us use $\text{Stack}_{\text{COMP}}(t)$ to denote the contents of M's pushdown store at time t in COMP and use $|\text{Stack}_{\text{COMP}}(t)|$ for its size. In the following proof it is vital that the size of the pushdown store does not grow exponentially large.

LEMMA 1. (a) *For all* t, $|\text{Stack}_{\text{COMP}}(t)| \leqslant O(n^2)$;

(b) $|\text{COMP}| \leqslant O(n^3)$.

Proof. (a) let q_t be the state, H_t be the vector of the head positions, and a_t the top stack symbol at ID_t. Let also $N = |Q|^2 |\Gamma| k^2(n+1)^2$. Suppose that $N' = |\text{Stack}_{\text{COMP}}(t_0)| > N$, for some t_0. For each $s = 0, 1, ..., N'$ we define

$$F(s) = (q_{t_1}, H_{t_1}, a_{t_1}, q_{t_2}, H_{t_2}),$$

where $t_1 = t_1(s)$ is the last time before t_0 and $t_2 = t_2(s)$ is the first time after t_0 such that $|\text{Stack}_{\text{COMP}}(t_1)| = |\text{Stack}_{\text{COMP}}(t_2)| = s$. (We can assume that M pushes and erases at most one symbol a time, and such t_1, t_2 must exist.) Then $F(s)$ takes on at most N values, because H_t can have no more than $k(n+1)$ values in COMP. Therefore $F(r) = F(s)$ for some $r < s$. But then we can cut out the sub-computation from $t_1(r)$ to $t_1(s)$ and from $t_2(s)$ to $t_2(r)$, obtaining a computation shorter than COMP.

(b) Immediate from Fact 3 and (a). ∎

We now adopt some ideas and definitions developed to prove lower bounds on string-matching [L1] and on Turing machines [L2, LV]. Consider the computation COMP of M on I.

We say a block w_i (or a sub-block w_{ij}) is *directly matched* if there is a time when one head of M stays in w_i (resp. w_{ij}) and another head of M simultaneously stays in the other w_i (resp. w_{ij}).

LEMMA 2. *There exists an* i, $1 \leqslant i \leqslant b$, *such that* w_i *is not directly matched.*

Proof. One pair of distinct heads obviously can directly match only one w_i. Since there are $\binom{k}{2}$ ($< b$) pairs of heads, some w_i cannot be directly matched. ∎

Now fix the i in Lemma 2 such that w_i is not directly matched, where $w_i = w_{i1} \cdots w_{im}$.

Let u, v be the two occurrences of w_{ij}. We say that w_{ij} is *indirectly matched* if there are times $t_1 < t_2$ such that:

im1. Some head h_p is in u (resp. v) and M makes a "push" at time t_1.

im2. Some head h_q is in v (resp. u) at time t_2 and $|\text{Stack}_{\text{COMP}}(t_2)| = |\text{Stack}_{\text{COMP}}(t_1)| + 1$.

im3. For each t, $t_1 < t < t_2$, $|\text{Stack}_{\text{COMP}}(t)| > |\text{Stack}_{\text{COMP}}(t_1)|$.

Conditions (im2) and (im3) ensure that at t_2 M sees exactly the symbol pushed at t_1. Without loss of generality, we assume that changing a top stack symbol is possible only by popping it and then pushing another.

LEMMA 3. *There exists a* j, $1 \leqslant j \leqslant m$, *such that* w_{ij} *is not indirectly matched.*

Proof. Consider any pair of heads (h_p, h_q), $p, q \in \{1, 2, ..., k\}$. We claim that h_p and h_q can cooperate to indiretly match only one sub-block. Without loss of generality, assume that h_p enters a w_i (it does not matter which one) first. Note that before h_p leaves w_i h_q cannot enter the other occurrence of w_i, since w_i is not directly matched.

Suppose two sub-blocks w_{ij} and $w_{ij'}$ were indirectly matched using h_p and h_q for $j < j'$. Let t_1 and t_2 be the times from the definition of indirect matching for w_{ij} and t'_1, t'_2 for $w_{ij'}$.

By the previous paragraph we have $t_1 < t'_1 < t_2 < t'_2$. By (im3) for $w_{ij'}$ we have

$$|\text{Stack}_{\text{COMP}}(t_2)| > |\text{Stack}_{\text{COMP}}(t'_1)|.$$

By (im3) for w_{ij} we have

$$|\text{Stack}_{\text{COMP}}(t'_1)| > |\text{Stack}_{\text{COMP}}(t_1)|.$$

The two inequalities above together imply that

$$|\text{Stack}_{\text{COMP}}(t_2)| > |\text{Stack}_{\text{COMP}}(t_1)| + 1,$$

a contradiction with (im2) for w_{ij}. ∎

LEMMA 4. *In computation COMP, each sub-block w_{ij} of the input must be either directly matched or indirectly matched.*

Proof. Suppose a sub-block w_{ij} is neither directly matched nor indirectly matched in the computation COMP. We shall show that W is not random by reconstructing w_{ij} with a small amount of information. Recall that a ID_t of COMP is a tuple (positions of input heads, state of the machine, top stack symbol, height of stack) which describes the status of M at time t.

Let u be the first occurrence and v the second occurrence of w_{ij}. The general idea of the proof is to record some information about the behavior of M on I, and then use this information to construct w_{ij}. Intuitively, COMP contains some time intervals when M may collect or use information about u, by either having a head in u or reading a part of the stack which was pushed while some head was in u. For each such interval we store only the first and last configurations. Having all this information we reconstruct w_{ij} as follows. For each word y of length $|v|$ we create an input I_p in which u is replaced by a sequence of 0's and v by y. For each such I_p we simulate M on I_p, skipping these parts of the computation which correspond to the information recorded before. If, during the simulation, all the transitions were consistent with the recorded information, and we reach an accepting configuration, then using the assumption that w_{ij} was not matched in COMP we can deduce that we have discovered an accepting computation of M on I_p, which implies that $y = w_{ij}$.

Consider COMP. We divide now all times in COMP into several subsets:

(1) *u-intervals.* These are the time intervals when some heads scan u. Let us denote them by $U_1, U_2, ..., U_a$, $a \leqslant k$.

(2) *v-intervals.* These are the time intervals when some head scans v.

(3) All remaining time intervals.

In (1) and (2) the intervals are chosen to be maximal. We have that all u-intervals and v-intervals are pairwise disjoint (because w_{ij} is not directly matched). Now by $R_t(U_s)$ we will denote the region of the stack containing the symbols pushed onto the stack at times in U_s, which are still on the stack at time t. For each t, $R_t(U_1)$, $R_t(U_2)$, ... are disjoint and continuous segments of the stack. Furthermore, $R_t(U_{s+1})$ is always above $R_t(U_s)$, if both are nonempty.

For each $s = 1, ..., a$ we store the following information about COMP:

(a) Let t_1 be the first time step in U_s. Record t_1 and ID_{t_1}. This needs $O(k \log n)$ bits, by Lemma 1.

(b) Let t_2 be the first time step after U_s, and c_s the size of $R_{t_2}(U_s)$. Record t_2, ID_{t_2}, and c_s. This needs $O(k \log n)$ bits.

(c) Let t_0, $t_0 < t_1$, be the time step when the symbol right below $R_{t_2}(U_s)$ was pushed onto the stack. Record t_0 and ID_{t_0}.

(d) Now we will define a sequence of times $\tau_1, \tau_1', \tau_2, \tau_2', ..., \tau_e, \tau_e'$. We take $\tau_1 = t_2$. Suppose that $\tau_1, \tau_1', ..., \tau_{d-1}', \tau_d$ are already defined for some $d \geqslant 1$. If $R_t(U_s)$ disappears before the next v-interval after τ_d (or if there are no more v-intervals) then τ_d' is the first t such that $R_t(U_s)$ is already empty. Otherwise τ_d' is such t before the next v-interval that $|\text{Stack}_{\text{COMP}}(t)|$ is minimized (in case of a tie take the last such t). And τ_{d+1} is the first time step after τ_d' such that

$$|\text{Stack}_{\text{COMP}}(\tau_{d+1})| = |\text{Stack}_{\text{COMP}}(\tau_d')| - 1.$$

We store each τ_d, τ_d', and ID_{τ_d}, $ID_{\tau_d'}$. This needs $O(k^2 \log n)$ bits.

Further we will write $t_l(s)$, $\tau_l(s)$, and $\tau_l'(s)$ to indicate that they are the times stored for s. The time intervals $[\tau_d(s), \tau_d'(s)]$ will be called *τ-intervals.* By the definition and by the assumption that w_{ij} is not indirectly matched, τ-intervals and v-intervals are disjoint. However, τ-intervals can overlap with u-intervals.

After recording all above information we are ready to construct our short program to generate w_{ij}. Notice that when we construct the information in (a)–(d), we use whole W; however, once we record down (a)–(d) we use only $W - w_{ij}$ plus information recorded in (a)–(d) in the program described below.

For every y such that $|y| = |w_{ij}|$, construct the following input using $W - w_{ij}$:

$$\begin{aligned} I_y = {} & \mathbb{¢} w_1 \# \cdots \# w_{i-1} \# w_{i1} \cdots w_{ij-1} 0^{|w_{ij}|} w_{ij+1} \cdots w_{im} \\ & \# w_{i+1} \# \cdots \# w_b \$ w_b \# \cdots \# w_{i+1} \\ & \# w_{i1} \cdots w_{ij-1} y w_{ij+1} \cdots w_{im} \# w_{i-1} \# \cdots \# w_1 \mathbb{¢}. \end{aligned}$$

Simulate M on I_y, in a *dovetailing* style (for all y's), as follows. If any inconsistency

occurs in the following simulation, try other nondeterministic choices. During the simulation we will skip over u- and τ-intervals using the information recorded above. It may happen that after skipping such an interval we are inside another u- or τ-interval T. In this case the computation should skip to the end of T. We repeat above until we are out of u- and τ-intervals. Then adjust the configuration appropriately.

Suppose that M is at time t. If t is not equal to $t_1(s)$ or $\tau_d(s)$, then honestly use the transition function of M. In addition, if $t=t_0(s)$, check the consistency of the status of M with $ID_{t_0(s)}$. If $t_0(s)<t<t_1(s)$, check whether the height of the stack does not decrease below the stack height at $t_0(s)$. Otherwise:

(i) if $t=t_1(s)$ then do the following:

— check the consistency of the status of M with $ID_{t_1(s)}$,

— erase anything above $|\text{Stack}_{\text{COMP}}(t_0(s))|$ from the stack,

— push 0^{c_s} onto the stack,

— go to $ID_{t_2(s)}$;

(ii) if $t=\tau_d(s)$ then do the following:

— check the consistency of the status of M with $ID_{\tau_d(s)}$,

— go to $ID_{\tau'_d(s)}$.

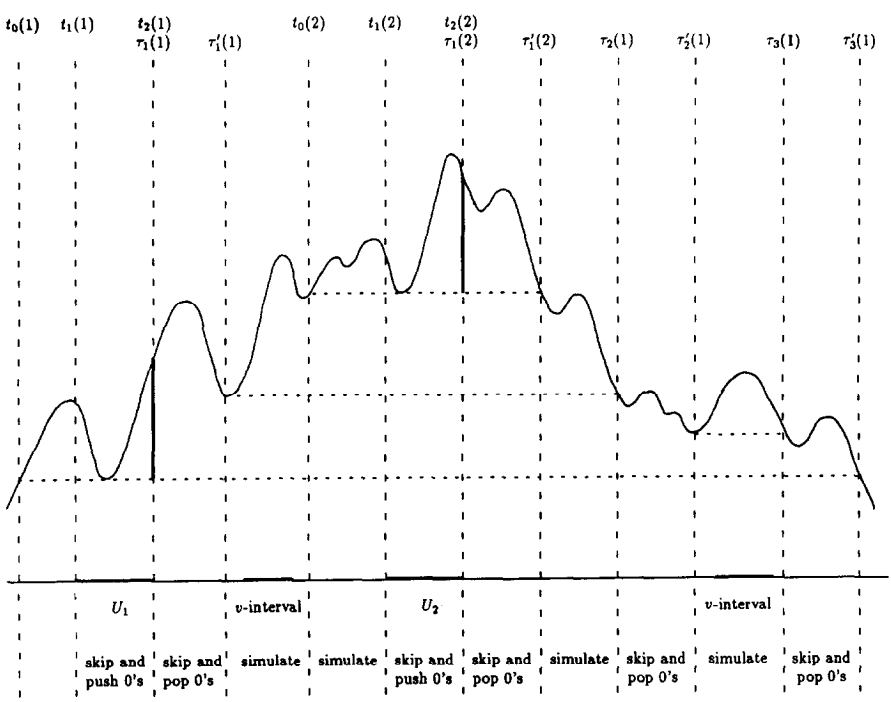

FIG. 1. An example of a simulation.

The instruction "goto ID_τ" consists of appropriately adjusting the input head positions, the state, the stack height (this amounts only to pushing or popping a number of 0's) and resetting t to τ. An example of a simulation is shown in Fig. 1. The vertical axis corresponds to the stack height, the horizontal axis to time. The darkened horizontal intervals denote u- and v-intervals. The darkened vertical intervals denote the segments $R_t(U_s)$. Now we will prove the following claim:

(*) M accepts I_y during the above simulation iff $y = w_{ij}$.

If $y = w_{ij}$ then COMP or some other accepting computation consistent with COMP will be found during the simulation. To prove the "only if" part, suppose that M accepts I_y and let C be an accepting computation for I_y found during the simulation. Actually, from the formal point of view, C is not a computation at all because it contains "gaps": from $ID_{t_1(s)}$ to $ID_{t_2(s)}$ and from $ID_{\tau_d(s)}$ to $ID_{\tau'_d(s)}$, for each d and s. Let I'_y be the input obtained from I by substituting the second w_{ij} by y. Fill each gap in C by a corresponding part of COMP and let it be C'. Each such gap corresponds to an u- or τ-interval. Since these intervals are disjoint with the v-intervals, none of the heads will enter y in C' at these time intervals. This implies that C' is a legal computation on I'_y. But C' is accepting. Therefore $y = w_{ij}$.

The amount of information (other than $W - w_{ij}$) used in above program is only $O(k^3 \log n)$ according to (a)–(d) above. So, by choosing large enough $|W|$, we have

$$K(w_{ij} \mid W - w_{ij}) < |w_{ij}|/2,$$

a contradiction. ▮ (Lemma 4)

Now by Lemmas 2 and 3, there is a w_{ij} which is neither directly matched nor indirectly matched. This contradicts Lemma 4. The proof of the main theorem is now complete. ▮ (Main Theorem)

Remark. A k-DPDA version of the above theorem would be much simpler since Lemma 1 and many other considerations are not needed.

3. Consequences

Theorem 2. *There is a language which is acceptable by a 3-FA but not k-DPDA for any k.*

Proof. As in [YR], consider language,

$$L' = \{w_1 \# \cdots \# w_n \$ w'_n \# \cdots \# w'_1 \mid w_i \neq w'_i \text{ for some } i\}.$$

A 3-FA accepts L' easily. However no k-DPDA can accept L', since otherwise $L'' = \overline{L'} \cap \{w_1 \# \cdots \# w_b \$ w'_b \# \cdots \# w'_1\}$ would be acceptable by a k-DPDA because the second component is regular for fixed b. However, by Theorem 1, L'' cannot be accepted by a k-DPDA for $b > \binom{k}{2}$. ▮

We can strengthen Theorem 2.

THEOREM 2'. *There is a language which is acceptable by a 2-FA but not k-DPDA for any k.*

Proof. As in [YR], consider language,

$$L^* = \{x_1 \¢ y_1^* x_2 \¢ y_2^* \cdots {}^* x_n \¢ y_n \#$$
$$x_n' \¢ y_n'^* x_{n-1}' \¢ y_{n-1}'^* \cdots {}^* x_1' \¢ y_1' \mid$$
$$\text{there exist } i \text{ and } j \text{ such that } x_i = x_j' \text{ and } y_i \neq y_j'\}.$$

Obviously, L^* can be accepted by a 2-FA which simply guesses i and j and does the matching. Now define

$$R_b = \{1 \¢ w_1^* 2 \¢ w_2^* \cdots {}^* b \¢ w_b \# b \¢ w_b'^* \cdots {}^* 2 \¢ w_2'^* 1 \¢ w_1'\},$$

where 1, 2, ... are represented in binary. Clearly, R_b is regular for any fixed b. Now if L^* can be accepted by a k-DPDA, then $\overline{L^*}$ can also be accepted by a k-DPDA, and so can $L''' = \overline{L^*} \cap R_b$. However, by the same proof of Theorem 1, L''' cannot be accepted by a k-DPDA for $b > \binom{k}{2}$. ∎

This improves some results in [YR] and [C]. We now prove another result which says that more heads cannot replace nondeterminism.

THEOREM 3. $\text{CFL} \not\subseteq \bigcup_k \text{DPDA}(k)$.

Proof. In L^* defined above change $x_i' \¢ y_i'$ to $(x_i' \¢ y_i')^R$ for every i. The resulting language is context-free. Also by the similar proof as in Theorem 2' (except for requiring $b > \binom{k}{2} + k^2$), we can show that L^* after the above change cannot be accepted by a k-DPDA. ∎

COROLLARY 1. *We have the following hierarchies:*

(1) $\text{PDA}(1) = \text{CFL} \subsetneq \text{PDA}(2) \subsetneq \text{PDA}(3) \subsetneq \cdots$

(2) $\text{DPDA}(1) = \text{DCFL} \subsetneq \text{DPDA}(2) \subsetneq \text{DPDA}(3) \subsetneq \cdots$

(3) [YR] $\text{FA}(1) = \text{Regular sets} \subsetneq \text{FA}(2) \subsetneq \text{FA}(3) \subsetneq \cdots$

(4) [YR] $\text{DFA}(1) = \text{Regular sets} \subsetneq \text{DFA}(2) \subsetneq \text{DFA}(3) \subsetneq \cdots$.

The last two consequences are the results of [YR]. (2) was proved by [C].

Remark. To summarize, the above theorems tell us the following: (a) nondeterminism and a pushdown store cannot substitute for a head: $\text{DFA}(k) \not\subseteq \text{PDA}(k-1)$; (b) heads and a pushdown store cannot simulate nondeterminism: $\text{CFL or FA}(2) \not\subseteq \bigcup_k \text{DPDA}(k)$.

In [HI, p. 462], Harrison and Ibarra proved several important consequences about closure properties of the languages accepted by k-PDAs. Now all those con-

sequences are true. We list them in the next corollary. The proofs, on the assumption that Theorem 1 is true, can be found in [HI]. (Some of the following were proved by [C].)

COROLLARY 2. (a) *For each* $k \geq 2$, PDA(k) (DPDA(k)) *is not closed under intersection and complementation (intersection and union).*

(b) *For each* $k \geq 2$, DPDA(k) *is not closed under the operations of concatenation, closure, and transposition.*

ACKNOWLEDGMENTS

We thank the referee for many suggestions which considerably improved the presentation of the paper. Thanks also to Don Beaver for corrections.

REFERENCES

[AHU] A. V. AHO, J. E. HOPCROFT, AND J. D. ULLMAN, Time and tape complexity of pushdown automaton languages, *Inform. and Control* **13** (1968), 186–206.

[C] M. CHROBAK, Hierarchies of one-way multihead automata languages, *in* "12th Int. Colloq. Automata, Lang. Programming, Nafplion, Greece, 1985," Lecture Notes in Comput. Sci. Vol. 194, pp. 101–110, Springer-Verlag, New York/Berlin, 1986; *Theoret. Comput. Sci.* **48** (1986), 153–181.

[C1] S. A. COOK, Characterizations of pushdown machines in terms of time-bounded computers, *J. Assoc. Comput. Mach.* **13** (1971), 4–18.

[F] R. FLOYD, Review 14, *Comput. Rev.* **9** (1968), 280.

[GKS] Z. GALIL, R. KANNAN, AND E. SZEMEREDI, On nontrivial separators for k-page graphs and simulations by nondeterministic one-tape Turing machines, *in* "Proceedings, 18th ACM Symp. Theory of Comput. 1986," pp. 39–49.

[HLS] J. HARTMANIS, P. LEWIS, AND R. STEARNS, Hierarchies of memory limited computations, *in* "Proceedings, 6th Symp. on Switching Circuit Theory and Logical Design, 1965," pp. 179–190.

[HI] M. A. HARRISON AND O. H. IBARRA, Multi-head and multi-tape pushdown automata, *Inform. and Control* **13** (1968), 433–470.

[HU] J. E. HOPCROFT AND J. D. ULLMAN, "Introduction to Automata Theory, Languages, and Computation," Addison-Wesley, Reading, MA, 1979.

[I1] O. H. IBARRA, "Generalizations of Pushdown Automata," Ph. D. thesis, University of California, Berkeley, 1967.

[I2] O. H. IBARRA, On two-way multihead automata, *J. Comput. System Sci.* **7** (1973), 28–37.

[IK] O. H. IBARRA AND C. E. KIM, On 3-head versus 2 head finite automata, *Acta Inform.* **4** (1975), 193–200.

[L1] M. LI, Lower bounds on string-matching, submitted for publication, 1984.

[L2] M. LI, Lower bounds in computational complexity, Ph. D. thesis, Cornell University, 1985.

[LLV] M. LI, L. LONGPRE, AND P. M. B. VITANYI, On the power of the queue, "Structure in Complexity Theory," Lecture Notes in Computer Science Vol. 223, Springer-Verlag, New York/Berlin, pp. 219–233, 1986.

[LV] M. LI AND P. M. B. VITANYI, Tape versus queue and stacks: The lower bounds, submitted for publication.

[M1] S. MIYANO, A hierarchy theorem for multihead stack-counter automata, *Acta Inform.* **17** (1982), 63–67.

[M2] S. MIYANO, Remarks on multihead pushdown automata and multihead stack automata, *J. Comput. and System Sciences* **27** (1983), 116–124.

[M3] B. MONIEN, Transformational methods and their application to complexity problems, *Acta Inform.* **6** (1976), 95–108; Corrigenda, *Acta Inform.* **8** (1977), 383–384.

[M4] B. MONIEN, Two-way multihead automata over a one-letter alphabet, RAIRO *Inform. Théor.* **14** (1980), 67–82.

[M5] W. MAASS, Combinatorial lower bound arguments for deterministic and nondeterministic one-tape Turing machines, *Trans. Amer. Math. Soc.* **292**, No. 2 (1985), 675–693.

[MS] M. MAASS AND G. SCHNITGER, An optimal lower bound for Turing machines with one work tape and a two-way input tape, "Structure in Complexity Theory," Lecture Notes in Computer Science Vol. 223, Springer-Verlag, New York/Berlin, pp. 249–264, 1986.

[N] C. G. NELSON, "One-Way Automata on Bounded Languages," TR 14–76, Harvard University, July 1976.

[P] W. PAUL, Kolmogorov complexity and lower bounds, *in* "2nd International Conference on Fundamentals of Computations Theory, 1979."

[P1] W. PAUL, On-line simulation of $k+1$ tapes by k tapes requires nonlinear time, *in* "Proceedings, 23rd IEEE Found. of Comput. Sci., 1982," pp. 53–56.

[PS] R. PATURI AND J. SIMON, Lower bounds on the time of probabilistic on-line simulations, *in* "Proceedings, 24th IEEE Found. of Comput. Sci., 1982," pp. 343.

[PSS] W. PAUL, J. SEIFERAS, AND J. SIMON, An information-theoretic approach to time bounds for on-line computations, *in* "Proceedings, 12th ACM Symp. Theory of Comput., 1980," pp. 357–367.

[Pi] T. F. PIATKOWSKI, "n-Head Finite State Machines," Ph. D. thesis, University of Michigan, 1963.

[R1] A. ROSENBERG, "Nonwriting Extensions of Finite Automata," Ph. D. thesis, Harvard University, 1965.

[R2] A. ROSENBERG, On multihead finite automata, *IBM J. Res. Develop.* **10** (1966), 388–394.

[RS] S. REICH AND G. SCHNITGER, Three applications of Kolmogorov-complexity, "Proceedings, 23rd IEEE Found. of Comput. Sci., 1982," pp. 45–52.

[SV] W. J. SAVITCH AND P. M. B. VITANYI, On the power of real-time two-way multihead finite automata with jumps, *Inform. Process Lett.* **19** (1984), 31–36.

[S] J. SEIFERAS, "Nondeterministic Time and Space Complexity Classes," Ph. D. thesis, MIT, 1974.

[S0] J. SEIFERAS, Relating refined space complexity classes, *J. Comput. System Sci.* **14** (1977), 100–129.

[S1] I. H. SUDBOROUGH, "Computation by Multi-Head Writing Finite Automata," Ph. D. thesis, Pennsylvania State University, 1974.

[S2] I. H. SUDBOROUGH, One-way multihead writing finite automata, *Inform. and Control* **30** (1976), 1–20; *in* "Proceedings, Found. of Comput. Sci., 1971."